Cover Page

Title: A Migratory Approach to Dynamic Replication in Large-Scale Distributed Systems.

Contact Author and Affiliation: Dr. Indranil Gupta, Assistant Professor, Dept. of Computer Science, University of Illinois, Urbana-Champaign, USA.

Contact Address:
Department of Computer Science
University of Illinois, Urbana-Champaign
201 N. Goodwin Ave.
Urbana, IL, 61801.
email: indy@cs.uiuc.edu
Ph: (217) 265-5517
Fax: (217) 265-6494

Abstract: Distributed replication forms a significant component in many distributed systems. We consider *migratory* solutions for replica *location* in a large-scale distributed system. Replica location strategies decide dynamically how many times an object is replicated and where it is placed, in order to ensure availability, attacker-resistance, and scalability. Most traditional replica location schemes are static and reactive to failures. This paper presents dynamic and migratory schemes for replica location. More specifically, we present a new class of probabilistic protocols called *endemic protocols*. By using analytical techniques borrowed from the study of non-linear systems, and through large-scale simulations, we show how an endemic protocol can be used in building a decentralized and persistent file storage service. The protocols continuously migrate a small number of replicas of each object through the host population. This means an attacker will not be able to predict the exact number and locations of all replicas of a object. Contrary to intuition, endemic protocols can provide good performance by generating only a constant amount of network traffic at each host. Endemic protocols are resistant to massive failures and host churn. Existence of even one residual replica of an object in the system causes the system to regenerate more replicas. Analytically, endemics have the potential to preserve an object for several human generations, much like the persistent survival of folklores and endemic diseases such as common cold. The protocols are also very simple to implement.

Keywords: Distributed Replication, Replica location, Endemic protocols, Scalability, Reliability, File Systems.

Number of pages in paper: 10 regular pages.

A Migratory Approach to Dynamic Replication in Large-Scale Distributed Systems

Indranil Gupta
Asst. Professor, Dept. of Computer Science
Univ. of Illinois (Urbana-Champaign)
email: indy@cs.uiuc.edu

Abstract

Distributed replication forms a significant component in many distributed systems. We consider *migratory* solutions for replica *location* in a large-scale distributed system. Replica location strategies decide dynamically how many times an object is replicated and where it is placed, in order to ensure availability, attacker-resistance, and scalability. Most traditional replica location schemes are static and reactive to failures. This paper presents dynamic and migratory schemes for replica location. More specifically, we present a new class of probabilistic protocols called *endemic protocols*. By using analytical techniques borrowed from the study of non-linear systems, and through large-scale simulations, we show how an endemic protocol can be used in building a decentralized and persistent file storage service. The protocols continuously migrate a small number of replicas of each object through the host population. This means an attacker will not be able to predict the exact number and locations of all replicas of an object. Contrary to intuition, endemic protocols can provide good performance by generating only a constant amount of network traffic at each host. Endemic protocols are resistant to massive failures and host churn. Existence of even one residual replica of an object in the system causes the system to regenerate more replicas. Analytically, endemics have the potential to preserve an object for several human generations, much like the persistent survival of folklores and endemic diseases such as common cold. The protocols are also very simple to implement.

Keywords: Distributed Replication, Replica location, Endemic protocols, Scalability, Reliability, File Systems.

1 Introduction

Replication of data in distributed systems has been studied for many years, e.g., databases [8, 10], email and file systems [16, 17], and more recently, in peer to peer systems [5]. In a distributed group of processes, the tasks of replication middleware can be broadly divided into (a) replica location (placement), and (b) replica management. For a given object, replica location decides which processes bear responsibility for storing replicas of a given object, e.g., in Clearinghouse, all servers in the system eventually receive a copy of each object ("total replication"), while in Pastry [15], only those processes that hash to a value close to the object's hash, store replicas ("partial replication"). Replica management deals with how replicas are accessed and updated in a consistent manner, e.g, active and passive replica management are well known techniques [6].

This paper focuses only on replica location, and specifically, on scalable and reliable solutions for partial replication. Location deals with deciding "how many" and "where" replicas are located. Most existing solutions in literature [15, 16] locate replicas *statically* and *reactively*, i.e., the subset of hosts selected to hold replicas of a given object does not change unless a special event happens, e.g., one of the hosts could crash.

However, reactive replica location strategies have several limitations in real-life large-scale distributed systems. They can be expensive in peer to peer systems containing millions of hosts, where a large fraction of hosts have short lifetimes (O(several minutes)) and rejoin multiple times (6.4 times/day as reported in the Overnet system [3]). One could restrict replicas to be stored only on hosts with at least a threshold availability [4], but this does not address the following issue. More importantly, from a security stand-point, static and reactive strategies allow an attacker to easily locate and attack all the individual replicas of an object. A malicious host could track the current replicas for a given object, and then subject each of them simultaneously to a kind of attack (e.g., a DOS attack on each host, sus-

tained until the host crashed, would suffice) in such way that all copies of the object are destroyed.

We are studying dynamic and migratory replication strategies because they can avoid the above drawbacks, and provide other interesting properties. These strategies proactively move replicas of the object among different hosts in the system. Thus, an attacker finds it difficult to locate all the replica hosts. Further, if the replication protocol is designed so that the number of replicas is tolerant to host failures, availability of the object will not be affected by either short host availability periods or massive failures in the system.

The problem can be expressed formally as follows:

Distributed Responsibility Migration Problem: At any point of time in a group G of processes, each process is either a *responsible* process, or not. Further,

`Safety:` The number of responsible processes in the system never becomes zero.

`Liveness:` If a process p is a responsible process at give (local) time t_0, there is a time $t > t_0$ at which process p ceases to be a responsible process.

`Fairness (optional):` Over a long time of running, each process in the system bears responsibility for an equal fraction of time.

For a distributed file system application, where each object is a file, each file has a responsibility migration protocol running on its behalf. At any time, the responsible processes in the protocol for a file are the only ones storing replicas of the file. `Safety` ensures that a file, once inserted, never disappears from the system. `Liveness` ensures that a responsible process eventually sheds its responsibility of storing the file forever [1]. Static and reactive strategies attempt to satisfy safety but neither liveness nor fairness.

Migrating the replicas of an object can be combined with strategies for replica management. Traditional replica management techniques such as active or passive replication can be used, although this will require the support of a weakly consistent protocol (e.g., SWIM [7, 9]) that always maintains transitive connectivity among the subgroup of processes that are marked as being responsible. Since the focus of this paper is on replica location, a discussion of the interaction with replica management is left to a separate publication.

Other Applications of Distributed Responsibility Migration: Besides the distributed file system application, we also note the applicability of responsibility migration algorithms for other coordination activities, e.g., the subgroup of responsible processes can be used to globally order multicasts, run a consensus protocol for the group, etc. These are beyond the scope of this paper.

The problem definition leads to the following impossibility result:

Theorem 1 *(Impossibility of achieving safety in an asynchronous network)***:** No migratory replication protocol can achieve `Safety` in an asynchronous setting. This is because there exists a run where at some instant of time, all responsible processes crash simultaneously.

In this paper, we present a novel probabilistic protocol for distributed responsibility migration that achieves liveness, fairness, and *probabilistic safety* (i.e., w.h.p., the number of responsible processes in the system never becomes zero). The behavior of the new protocol bears resemblance to the persistent survival of folklores in human society, and common cold in human populations, both for centuries and in a manner that resists the death of individuals. Hence we term the new protocols as *Endemic Protocols*.

Contrary to intuition, endemic protocols can offer scalable and reliable behavior w.r.t. liveness, safety, and fairness, while incurring only a constant message overhead at each process in the group.

It is also possible to use endemic protocols in the design of distributed spyware that avoid detection, or by writers of computer viruses. Hence it is important for us to understand the dynamics associated with these protocols.

A word on the analysis style in this paper. The behavior of a processes running an endemic protocols can be expressed as a differential equation. Hence our analysis borrows techniques from the classical study of non-linear differential equations [18].

The rest of this paper is organized as follows. Section 2 discusses related work and the system model. Section 3 describes the endemic protocol and Section 4 presents a formal analysis. Section 5 presents simulation results from a prototype implementation. We conclude in Section 6.

[1] This does not preclude the process from becoming responsible again at some later point of time.

2 Related Work, and Model

Related Work: Several textbooks, including the one by Colouris et al [6], provide discussion of classical replication used in distributed systems. There has also been significant recent work. Cohen and Shenker [5] have proposed proactive replication schemes for partial replication of files in peer to peer systems that utilize "blind search". Yu and Vahdat [20] study the tradeoffs between consistency and availability. Demers et al proposed an epidemic algorithm for complete replication in a database system [8]. Gray et al [10] argue against the dangers of large-scale replication and warn that maintaining consistency among a large number of replicas might counter the goal of scalability. Our paper considers only small numbers of replicas. Ratner et al considered optimistic partial replication in the Ficus system [14], where the replica nodes of each object are organized in a virtual ring. The paper does not discuss scalable replica location strategies. Ranganathan et al [13] consider replication strategies for Grid applications, where location and management are cleanly separated.

R. Anderson, in his 1996 paper [1] outlined an "Eternity Service" for storage of files across a distributed system of hosts. The implicit challenge was "(to design software for a system where) your file, once posted on the service, cannot be deleted. As you cannot delete it yourself, you cannot be forced to delete it, either by abuse of process or by gun at your wife's head." Several scalable decentralized solutions have been proposed towards this problem, e.g., Oceanstore [11], LOCKSS[12], PAST[15]

An endemic protocol can be seen as a special case of a probabilistic I/O automata [19]. Yet, we are not aware of papers that analyze probabilistic protocols through the use of differential equations. Finally, our analysis of endemic protocols bears resemblance to the analysis of endemic diseases in epidemiology (e.g., [2]), yet our treatment is motivated by algorithmics and practical considerations.

System Model: We assume a closed group G of N processes, communicating over an asynchronous network. Each process may suffer from crash-stop or crash-recovery failures. The closed group assumption means that there are no joins by new processes. We verify that our protocols perform well in open goups, in Section 5. The communication medium is unreliable, and can drop messages or connections. Each process also knows about the maximal group membership, i.e., the other $N-1$ processes [2]. Our protocol analysis assumes that the group size N is large enough so that variables that denote the number of processes in a given state of a finite state machine can be assumed to be continuous. This also enables us to assume that these variables vary continuously in time (as opposed to discretely).

3 Migratory Replication

For concreteness, our discussion directly addresses the scenario where the endemic protocol is used to implement a distributed file system. In other words, the *objects* that are being replicated will be individual files. In other applications, the object might simply be a (replicated) token, giving the holder responsibility. We also restrict our discussion with respect to a single object only.

3.1 A Simple Solution, and its Drawback

Consider a replica migration protocol where a process storing an object replica hands it off to another process after a while and immediately deletes the object. A crash-stop failure of the former process before the transfer effectively destroys a object replica. Over time, the number of replicas of a given object will then go down to zero (unless there is a periodic refresh). Thus, this algorithm satisfies `Liveness`, but not `Safety`.

3.2 An Endemic Protocol for Migratory Replication

We describe the endemic protocol for the responsibility migration problem. At any time, a given process is in one of three states with respect to a given object. If the process is responsible and is storing a replica of the object, it is in the state *Stash*. If the process is not responsible, it could be in one of two states. The process might be unwilling to store the object, placing itself in an *Averse* state. Finally, the process might be in a *Receptive* state, meaning that it is not currently storing a copy of the object, but is willing to do so. In other words, a process is *responsible* if and only if it is in the Stash state.

State actions and transitions are probabilistic. (i) A process p in the Receptive state periodically

[2] Well-known results can be used to reduce this size to logarithmic.

chooses b targets (b is a small constant) uniformly at random, and if any of these targets is in the Stash state, process p changes its state to Stash (after an object transfer). (ii) A process p in the Stash state periodically chooses b processes uniformly at random; any of the selected target processes that is Receptive immediately transitions to the Stash state (after an object transfer). (iii) A process p in the Stash state also periodically tosses a coin with a biased heads probability γ ($\in [0,1]$) - if the coin falls heads, process p changes its state to Averse. This transition is accompanied by a deletion of the object replica at p. (iv) Finally, a process p in the Averse state periodically tosses a coin with heads probability α ($\in [0,1]$), and changes state to Receptive if this coin falls heads up.

The above protocol is shown in Figure 1. Each action is performed once at the start of each protocol period. Protocol period lengths are fixed at all processes, but their start times are arbitrary across processes. Our analytical results hold for the average protocol period duration across the group. Notice that the protocol requires neither global clocks nor global synchronization or agreement.

For the distributed file system application in a group of hosts, the endemic protocol can be implemented by running two threads per host - a daemon listening on a standard port for incoming connections, and another that wakes up periodically. We also assume that for an object, the (unique) owning host of the object actively maintains a list of current object replica holders. This requires each host to contact the owner of an object when the host makes a transition into or out of the Stash state [3].

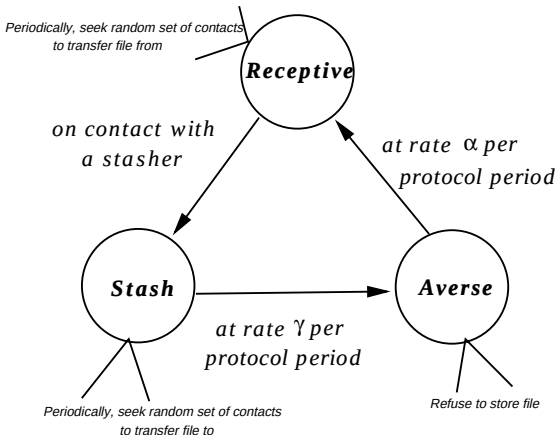

Figure 1: **A Simple Endemic Protocol:** A process is in one of three states with respect to a object. It is storing the object only in the Stash state. In the Stash (respectively the Receptive) state, it periodically selects a small set of processes to contact, seeking to transfer the object to (resp. from) each remote process. In the Averse state, it refuses to accept any transfers to it for the object. The transitions from states Stash to Averse and Averse to Receptive happen with probabilities γ and α per round respectively.

4 Endemic Protocol Analysis

First, we note that the endemic protocol of Figure 1 satisfies `Liveness` if $\gamma > 0$. Since the protocol is symmetric across processes, it also satisfies `Fairness`. The probabilistic safety of the protocol is discussed at the end of this section.

Let x, y, z respectively be the numbers of receptives, stashers and averse processes in the system at a given point of time. We analyze the endemic replication protocol from the previous section in a static group where $x + y + z = N$ always, N being a large constant. Our experiments vary N [4].

If x, y, z are assumed to be continuous variables, Similar to [2], we assume x, y, z to have real values, and vary continuously over time. We can write the rates of change of x, y, z as:

$$\begin{aligned} \dot{x} &= -\beta xy + \alpha z \\ \dot{y} &= \beta xy - \gamma y \\ \dot{z} &= \gamma y - \alpha z \end{aligned}$$

Here, β is the "successful contact rate" between a stasher and a receptive, per protocol period. Recall that each stasher or receptive selects b processes uniformly at random during each protocol period. If a fraction s of started object transfers complete successfully, the successful contact rate can be calculated as $1-(1-\frac{b}{N}s)^2$, which reduces to $\frac{2b'}{N}-(\frac{b'}{N})^2$, where $b' = bs$. For large N, this gives $\beta = \frac{2b}{N}$. For simplicity, we shall assume henceforth that $s = 1$, meaning that $b = b'$.

Equilibrium occurs when $(\dot{x}, \dot{y}, \dot{z}) = 0$. Substituting in the above equations gives us two equilibria:
$(x_\infty, y_\infty, z_\infty) = (N, 0, 0)$
$(x_\infty, y_\infty, z_\infty) = (\gamma/\beta, \frac{N-\gamma/\beta}{1+\gamma/\alpha}, \frac{N-\gamma/\beta}{1+\alpha/\gamma})$ (2)

The first equilibrium can result if all copies of the object disappear from the group. If the number of stashers thus becomes zero, the object has disappeared from the system. Subsequently, all processes

[3] As our experiments show, this has low overhead.

[4] For the purposes of the analysis, we shall also assume that all processes have equal clock speeds; this is not required by the actual protocol.

will eventually transition to the Receptive state.

The second equilibrium is more desirable, since it guarantees `Safety`. We are thus generally interested in finding out which of these two equilibrium points is *stable*. In other words, given an initial state that contains an arbitrary combination of stashers, receptives, and averse processes, how does the group of processes behave? Which of these two equilibria does it move towards? How quickly does it move? Is the protocol self-correcting, i.e., do small displacements from the desirable equilibrium point die out? For what values of α, β, γ is this true?

These questions would be easy to answer if we could solve the system of differential equations (1). However, an explicit solution is difficult in the case of this non-linear system of equations. Hence, we resort to the style of analysis used in the study of non-linear dynamics [18].

4.1 Are the Equilibria Self-Correcting?

We study this through a perturbation analysis. Let us assume a small deviation from the equilibrium point. Let us start the system in the state:

$$(x_0, y_0, z_0) = (x_\infty(1+u), y_\infty(1+v), z_\infty(1+w))$$

where $0 \leq u, v, w \ll 1$. Substituting this into equations (1), we get:

$\frac{\gamma}{\beta}\dot{u} = -\beta\frac{\gamma}{\beta}\frac{N-\gamma/\beta}{1+\gamma/\alpha}(1+u)(1+v) + \alpha\frac{N-\gamma/\beta}{1+\alpha/\gamma}(1+w)$

$$\text{or } \dot{u} = \frac{\beta N-\gamma}{1+\gamma/\alpha}(w-u-v) \quad (3a)$$

where the uv term is ignored as $u, v \ll 1$. Similarly,

$$\dot{v} = \gamma u \quad (3b)$$

$$\dot{w} = \alpha(v-w) \quad (3c)$$

where we have assumed that $y_\infty = \frac{N-\gamma/\beta}{1+\gamma/\alpha} \neq 0$. Eliminating w and v from equations (3a-c), we get the following differential equation for u.

$$\frac{1}{\sigma\alpha}\ddot{u} + \dot{u}(\frac{1}{\sigma}+\frac{1}{\alpha}) + u(1+\frac{\gamma}{\alpha}) = 0.$$

This can be written as a system of two linear differential equations by introducing a new variable $t = \dot{u}$. In matrix form, the equations are

$$\dot{T} = A \cdot T, \quad (4)$$

where

$$T = \begin{bmatrix} t \\ u \end{bmatrix}, \qquad \text{A} = \begin{bmatrix} -(\sigma+\alpha) & -\sigma(\gamma+\alpha) \\ 1 & 0 \end{bmatrix}$$

Lemma 1: The only equilibrium point in equation system (4) is $(t, u) = (0, 0)$.

Proof: It is easy to see that $\dot{u} = 0$ implies $t = 0$, and then $\dot{t} = 0$ gives $u = 0$.

Corollary: The equation system $(3(a, b, c))$ has only one equilibrium point $(u, v, w) = (0, 0, 0)$. This is easy to see by substitution of the equilibrium value $u = 0$ into equations $(3(b, c))$.

Theorem 2 *(All paths lead to the Second Equilibrium)*: For the system of differential equations (1), the equilibrium point $(x_\infty, y_\infty, z_\infty) = (\gamma/\beta, \frac{N-\gamma/\beta}{1+\gamma/\alpha}, \frac{N-\gamma/\beta}{1+\alpha/\gamma})$ is *always stable*, given that $\alpha, \gamma > 0$ and $N > \frac{\gamma}{\beta}$.

Proof: The behavior of the perturbation (u, v, w) is given by equation (4), and this has one equilibrium point of $u = (v = w =)0$. The stability of this equilibrium point depends on the trace and determinant of matrix A. The theory of linear differential equation systems tells us [18] that if the trace is negative and the determinant is positive, the equilibrium point is stable. However, if the trace and determinant were both positive, the equilibrium would be unstable. Finally, if the determinant is negative, the equilibrium is a saddle point (i.e., some trajectories in the vicinity converge to it, and the rest diverge).

We can calculate:

$\tau = \text{trace}(A) = \text{sum of leading diagonal elements in } A = -(\sigma+\alpha)$, and

$$\Delta = \det(A) = \sigma(\gamma+\alpha) \quad (5)$$

By our choice of parameters, $\sigma = \frac{\beta N-\gamma}{1+\gamma/\alpha} > 0$ and $\alpha, \gamma > 0$. Therefore, we *always have that* $\tau < 0, \Delta > 0$. [18] then tells us that the solution of $u = 0$ for equations (4) is *always stable*.

From equations $(3(b, c)$, Lemma 1 and the corollary, we can say that any perturbations $((u, v, w)$ in the values of $(x, y, z))$ around the second non-trivial equilibrium point will die out, and the system converges back. □

Corollary: If $N < \frac{\gamma}{\beta}$ and $\alpha, \gamma > 0$, then given that $x, y, z > 0$, the system of equations (1) has the (stable) equilibrium point $(N, 0, 0)$.

Lemma 2 *(First Equilibrium is a Saddle Point)*: If $\alpha, \gamma, \sigma > 0$ and $N > \frac{\gamma}{\beta}$, the first equilibrium point $(x, y, z) = (N, 0, 0)$ is a *saddle point*. This means that it is partly stable; as long as y remains 0, the system converges back to the equilibrium. How-

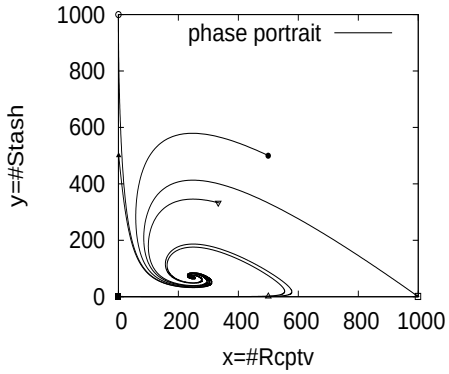

Figure 2: **Stability of Non-trivial Equilibrium Point - Stable Spiral:** The plot shows the phase portrait obtained by simultaneously plotting (x, y) over time. This plot shows the system started at these initial points - (x, y, z)=blank square (999,1,0), dark square (0,1,999), blank circle (0,1000,0), dark circle (500,500,0), blank triangle (500,1,499), dark triangle (1,500,499), and blank inverted triangle (333,333,334). For the parameter setting $N = 1000, \alpha = 0.01, \beta = 0.004, \gamma = 1.0$, the non-trivial equilibrium point above is a stable spiral.

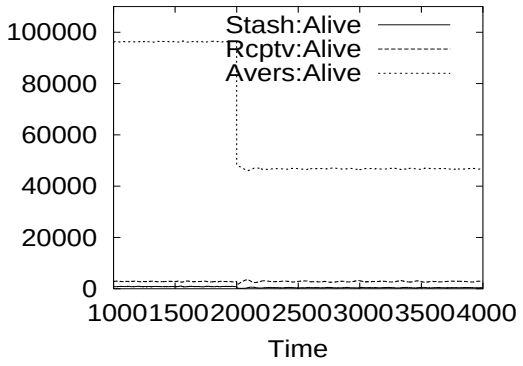

Figure 3: **Fault-tolerance A: Number of stashers, receptives, averse.** Massive system failure (of 50% of the hosts) at time t=2000 in a 100,000 host system does not cause instability in the system.

ever, inclusion of even a single stasher will drive the system towards the second, more stable equilibrium with a larger, non-zero number of replicas for the object.

4.2 Convergence Time to Equilibrium

The nature of the trajectories around the second stable equilibrium point, as well as the time taken by the protocol for convergence, both depend on the eigenvalues and eigenvectors of the matrix A defined in the last section. The eigenvectors of A are $\lambda_1 = \frac{\tau+\sqrt{\tau^2-4\Delta}}{2}$, and $\lambda_2 = \frac{\tau-\sqrt{\tau^2-4\Delta}}{2}$.

From equations (5) in the last section, we can calculate:

$\tau^2 - 4\Delta = (\frac{\beta N-\gamma}{1+\gamma/\alpha} - \alpha)^2 - 4\frac{\beta N-\gamma}{1+\gamma/\alpha}\gamma$

Three cases arise:

1. $\tau^2 - 4\Delta < 0$ (eigenvalues distinct and complex): The variation of the displacement u in the number of susceptibles x, as a function of time, can be calculated as:

$u = u_0 e^{-\frac{t(\sigma+\alpha)}{2}} cos(t\sqrt{\sigma\gamma - \frac{(\sigma-\alpha)^2}{4}})$

where u_0 is the initial value of u. Notice that with time, u decreases exponentially fast to 0. The cosine term causes a (damped) oscillation in the value of u. This leads to a *stable spiral*, which means that the convergence takes the form of a damped oscillation.

2. $\tau^2 - 4\Delta > 0$ (eigenvalues distinct and real): The variation of u as a function of time is given by

$u = \frac{\dot{u_0}-\lambda_2 u_0}{\lambda_1-\lambda_2} e^{t\lambda_1} + \frac{\dot{u_0}-\lambda_1 u_0}{\lambda_2-\lambda_1} e^{t\lambda_2}$

where u_0 and $\dot{u_0}$ are the initial values of u and $\dot{u}$ respectively.

3. $\tau^2 - 4\Delta = 0$ (eigenvalues equal and real): The variation of u as a function of time can be calculated as $u = u_0 e^{-t\frac{\sigma+\alpha}{2}}$, where u_0 is the initial value of u.

The above discussion reveals that no matter what the values of α, β, γ (> 0), an initial displacement from the equilibrium point decreases exponentially quickly. Thus, the system is self-stabilizing.

Figure 2 illustrates, through examples, one of the types of equilibrium points possible - stable spirals. Such plots to show the variation of the three variables (x, y, z) over time are called *phase portraits* [18].

4.3 Probabilistic `Safety`: Longevity of Object Replicas

In any computer system, there is always a non-zero probability of all replicas of a object disappearing completely. We present a back of the envelope calculation of the likelihood of this happening in an endemic protocol that is at equilibrium. Each of the y_∞ stashers creates new stashers at a rate $\beta x_\infty 1 = \gamma$. Each stasher is also turning averse at the same rate γ, thus it is equally likely to die before creating any new stashers. The likelihood that none of the y_∞ stashers create any new replicas is $= (\frac{1}{2})^{y_\infty}$.

If protocol parameters α, γ, b are chosen so that $y_\infty = clog_2 N$, the probability of all stashers dying before creating new ones is $\frac{1}{N^c}$. If a protocol period is 6 minutes long, $N = 1024$ and 50 replicas gives us an expected object longevity of 1.28×10^{10} years. With $N = 2^{20}$ and 100 replicas, we get an object lifetime of 1.45×10^{25} years.

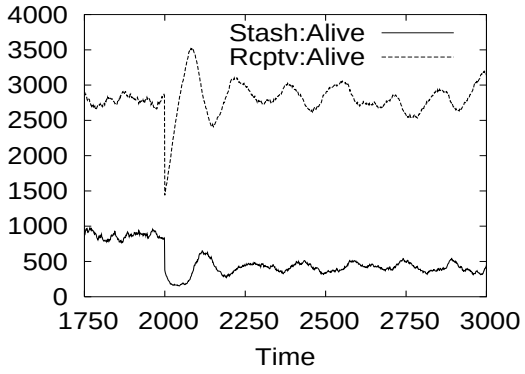

Figure 4: **Fault-tolerance B:** Zoomed-in version of Figure 3.

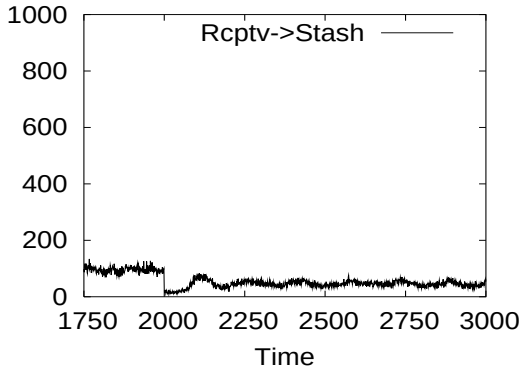

Figure 5: **Fault-tolerance C: File Flux Rate = number of file transfers per protocol period.** Occurrence of a massive number of failures at time $t = 2000$ does not affect file flux rate drastically.

5 Feasibility of an Implementation: A Simulation Study

We now present initial results from a C implementation of an endemic protocol, designed for the application of a distributed file system for persistent storage of files. The protocol was tested in a simulated environment, with multiple instances running synchronously over a simulated network, all on a single machine (1.7 Ghz Intel Celeron CPU, 256 MB RAM, WinXP Pro). Owing to the simplicity of the implementation, we are able to report numbers in 100,000-host groups.

Some implementation details are in order. The Mersenne Twister pseudorandom generator is used for random number generation. The protocol period is fixed at 6 minutes at each host. Each host deletes a file it is holding at a rate of γ (default=0.1) per period. Hosts remember for a while names of files they have recently deleted (making the host Averse), but forget such names at a rate of α (default=0.001) per period (after which they will be Receptive). During each period that a host is receptive towards (or is stashing) a file, it selects $b = 2$ other hosts uniformly at random from across the group and tries to transfer a file replica from (resp. to) the remote host. If a remote host is down, no exchange takes place - thus, not all "contacts" between a receptive and a stasher may lead to a file transfer.

In all the plots, the "Time" variable on the horizontal axis is normalized in protocol periods (6 minute intervals).

5.1 Fault-tolerance, Self-Survivability

A 100,000 host system initially at equilibrium is subject to failure of a random 50% of the hosts. Figure 3 and Figure 4 show that after the failure at time $t = 2000$, the number of averse and the number of stashers each drop by a factor of about two. The number of receptives does not change since after the failure, 50% of the contacts initiated by any alive host are directed at a crashed host, and are hence fruitless (this reduces the effective value of b by 2, thus retaining the original equilibrium value of $x_\infty = \frac{\gamma}{\beta}$). Figure 4 also shows that there is no wild variation in the number of stashers in spite of the massive failure, and that this number converges to the equilibrium value rather quickly.

The other feature revealed by Figure 4 is an oscillation in the numbers of stashers and receptives in the system around the respective equilibrium points. These oscillations are due to the random choice of contacts in the endemic protocol, but they die out. The frequency of the oscillations are more sluggish after the massive failure because fewer contacts are fruitful.

Notice that the behavior at times $t > 2500$ is also characteristic of a *heterogeneous* setting, where half the hosts are chronically averse to storing the file or even perhaps to running the protocol.

The self-correcting property of the endemic protocol arises from a constant flux of file replicas in the system. Figure 5 shows the number of hosts acquiring a new replica of a given file (turning stasher from receptive).

Reality Check: In a 100,000-strong system, the host would be storing a given file for 1% of the time (since number of stashers $\simeq 1000$), effectively 1 hour every 100 hundred hours, or a little over four days. The file would be stored for an average duration of 1 hour each time (expected time for a stasher to

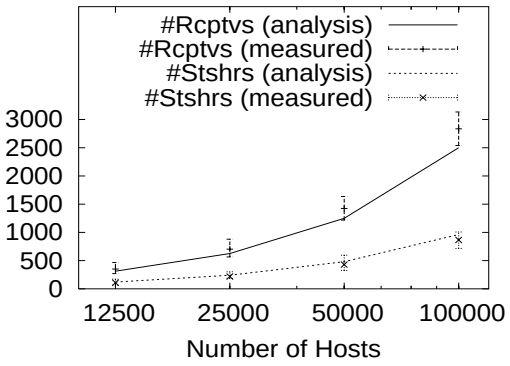

Figure 6: **Accuracy of Continuous Time Analysis:** The experimental results from the simulation match well with those predicted by the mathematical analysis. With $b = 2, \gamma = 0.1, \alpha = 0.001$, the above plot shows the median number (over a time interval 2000 periods long) and the analytically expected numbers of both receptives and stashers. The minimum and maximum measured values over this interval are also shown.

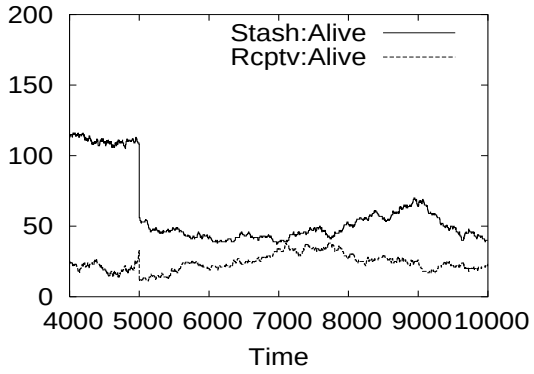

Figure 7: **Lowering the Number of Replicas A:** In a setting with $\alpha = 10^{-6}, \gamma = 10^{-3}$, the number of stashers and replicas in a 100, 0000 host system is very small. Massive failure of 50% of the hosts at time t=5000 causes the system to stabilize quickly. However, we do notice that the stabilization is more sluggish than in earlier settings with higher values of α, γ.

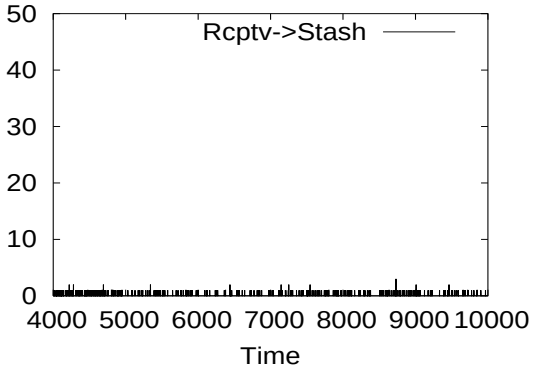

Figure 8: **Lowering the Number of Replicas B:** The total number of file replicas transferred across the system stays very small. Notice that the massive failure occurring at t=5000 does not affect the file flux rate.

turn averse is $\frac{1}{\gamma} = 10$ periods). During this hour, an average of one file transfer would be initiated by a remote host from this host (number of file transfers per period $\simeq 100$). A file size distribution such as that referred to in [15] (average file size is 88.2 KB) would mean that per file in the system, each host sees a transfer rate of 88.2 KB $\times 2/100$ hours = 3.92 bps.

5.2 Analysis vs. Real Behavior

Our analysis with differential equations in Section 4 assumed that the number of stashers, receptives, and averse take on continuous real values, and vary continuously over time. We compare those analytical results with numbers from a (discrete) simulation of the endemic protocol. Figure 6 shows the numbers of stashers and receptives (minimum, median, maximum) measured over a 2000 period duration (when the group is at equilibrium), and compares these numbers with those predicted by the mathematical analysis. We observe that the two tally quite closely, allowing us to verify the authenticity of the original analysis.

5.3 Lowering the Number of Replicas

The settings in our experiments described earlier in this section resulted in an average 1000 stashers and a file flux rate of 100 replicas / period in a 100K-host system. These numbers can be reduced by choosing smaller parameter settings. Figures 7 and 8 show that in a 100K system, choosing $\alpha = 10^{-6}, \gamma = 10^{-3}$ results in 100 replicas of a file at any given point of time, and a file transfer rate as low as 1 or 2 per period.

If each protocol period were 6 minutes long, these settings imply a bandwidth utilization (assuming an average file size of 88.2 KB) of 3.92×10^{-3} bps per file per host.

5.4 Untraceability of Replicas

Figure 9 shows the distribution of stashers over time in a population of $N = 1000$ hosts, with protocol parameters $b = 2, \gamma = 0.1, \alpha = 0.001$. The stable number of stashers is 88.63. The distribution of the stashers (dark dots) does not appear to have correlations either in time or across the host id. Thus, the plot reveals the following properties of the protocol: (a) **Replica Untraceability:** Replicas of a file are distributed across a random subset (and number) of hosts, and this distribution varies with time. This

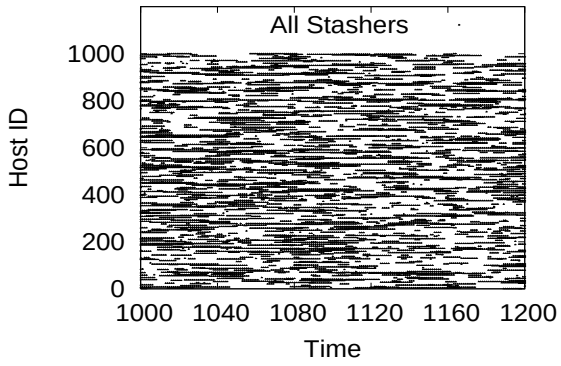

Figure 9: **Replica Untraceability and Load Balancing:** For a population of $N = 1000$ hosts, the plot shows which hosts are stashers at the end of every protocol period. The absence of significant horizontal lines indicates good load balancing (no hosts store a replica for very long). The absence of any correlations w.r.t. time or hostid in the figure shows the difficulty faced by an attacker who seeks to destroy all replicas of the file.

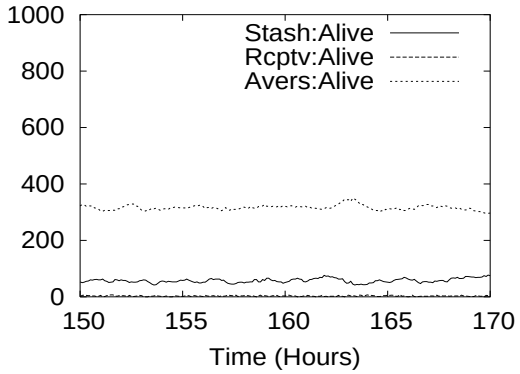

Figure 10: **Effect of Host Churn A:** Churn traces were injected hourly into the system with the protocol period set to 6 minutes at each host. This plot shows that the numbers of stashers, averse and receptives remain stable in the system, and the the number of stashers stays low.

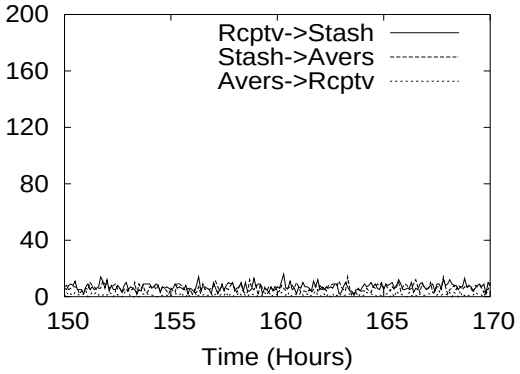

Figure 11: **Effect of Host Churn B:** For the experiment in Figure 10, this plot shows the number of state transitions, per protocol period, across hosts.

makes it extremely difficult for an attacker to guess accurately the location of all replicas of a file. In fact, unless the attacker knows about the location of all the replicas of a file at a given point of time *and* destroys all these copies before any new stashers are created (with the current parameters, one stasher is created every 40.6 seconds), the file will survive inside the system.

(b) **Load Balancing:** No hosts are unnecessarily overloaded, since dark lines appear to be uniformly distributed vertically.

5.5 Trace-based simulations: Effect of Host Churn

We study the behavior of endemic protocols in an *open group* where new processes are allowed to arrive into the system. Besides withstanding massive host failures, distributed protocols are also required to tolerate dynamic and continuous stresses. Deployed peer-to-peer systems for file sharing are known to face one such kind of dynamic stress known as "churning of hosts" - continuous arrival and failure of hosts. High churn rates have the potential to disrupt system-wide performance for long periods.

We investigate the behavior of the endemic protocol when subject to the kinds of host availability seen in the Overnet p2p system (availability traces obtained from the authors of [3]). Due to the limited sizes of these available traces, our system was restricted to a maximum of 2000 hosts. The original traces were taken at 1 hour intervals, but were smoothed over each hour for our experiments. Typical rates of churn observed from the traces range from 10% to 25% of the system size per hour.

We assume the worst case behavior w.r.t. disconnecting hosts, i.e., they delete *all* file replicas they are holding when they leave the system. When they rejoin the system, they are in the receptive state towards all files but do not participate in any startup file transfers.

The protocol period was set to 6 minutes, and the values of $\alpha = 0.005, \gamma = 0.1, b = 32$ used.

Figures 10 and 11 show the system behavior when the churning of nodes from each (hourly) trace file was spread out evenly throughout the hour. These plots also show the low file flux rate and stability of number of stashers in the face of churn.

6 Conclusion

In this paper, we have shown how a simple endemic protocol could be used to implement dynamic and migratory replica location in a large-scale distributed system. The discussion has been in the context of implementing a stable, forever, probabilistically indestructible and low cost solution towards forever

storage of files in a distributed setting. The protocol specifies three states - receptive, stash and averse - for each host w.r.t. each file in the system. It also specifies (probabilistic) transitions between these states. Through the use of analytical techniques traditionally used in the study of infectious diseases and non-linear dynamical systems, we are able to show that the endemic protocol is self-correcting and converges back to the equilibrium quickly even after massive failures. The protocol is also well-behaved under the kinds of host churn seen in today's peer to peer systems. The protocol has a low network bandwidth usage, and migrates replicas of a file across the population of hosts, making it difficult for an attacker to track and delete all copies of a file.

Continuing Work: We are building a distributed file system based on endemic protocols. The design issues deal with bridging ideas in this paper with replica management, e.g., coordinating replica holders to ensure file consistency.

References

[1] R. J. Anderson, "The Eternity Service" *Proc. 1st Intl. Conf. on the Theory and Applications of Cryptology*, Prague, Czech Republic, 1996.

[2] N. T. J. Bailey, "Epidemic Theory of Infectious Diseases and its Applications", Hafner Press, Second Edition, 1975.

[3] R. Bhagwan, S. Savage, G.M. Voelker, "Understanding availability", *Proc. 2^{nd} International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, pp. 135-140, Feb 2003.

[4] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total Recall: system support for automated availability management", *Proc. Usenix NSDI*, 2004.

[5] E. Cohen, S. Shenker, "Replication strategies in unstructured peer-to-peer networks", *Proc. ACM SIGCOMM Conference*, pp. 177-190, 2002.

[6] G. Colouris, J. Dollimore, T. Kindberg, *Distributed Systems: Concepts and Design*, Addison-Wesley, Third Edition, 2001.

[7] A. Das, I. Gupta, A. Motivala, "SWIM: Scalable Weakly-consistent Infection-style process group Membership protocol", *Proc. 2002 International Conference on Dependable Systems and Networks (DSN)*, pp. 303-312, 2002.

[8] A. Demers, D. H. Greene, J. Hauser, W. Irish, J. Larson, "Epidemic algorithms for replicated database maintenance", *Proc. 6^{th} ACM Symp. Principles of Distributed Computing (PODC 87)*, pp. 1-12, 1987.

[9] I. Gupta, T.D. Chandra, G.S. Goldszmidt, "On scalable and efficient distributed failure detectors", *Proc. ACM PODC*, pp. 170-179, 2001.

[10] J. Gray, P. Helland, P. O'Neil, D. Shasha, "The dangers of replication and a solution", *Proc. ACM SIGMOD Conference*, pp. 173-182, 1996.

[11] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao, "Oceanstore: an architecture for globalscale persistent store". *Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX,00)*, Cambridge, MA, Nov. 2000.

[12] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, M. Baker, Y. Muliadi, "Preserving peer replicas by rate-limited sampled voting", *Proc. 19th ACM Symposium on Operating Systems Principles (SOSP 03)*, Lake George, NY, Oct. 2003.

[13] K. Ranganathan, I. Foster, "Design and evaluation of dynamic replication strategies for a high performance data grid", *Proc. Intnl. Conf. Computing in High Energy and Nuclear Physics*, Beijing, Sep. 2001.

[14] D. Ratner, G. J. Popek, P. Peiher, "Peer replication with selective control", *Proc. MDA*, Hong Kong, 1999.

[15] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP 01)*, Banff, Canada, Oct. 2001.

[16] Y. Saito, C. Karamanolis, M. Karlsson, M. Mahalingam, "Taming aggressive replication in the Pangaea file system", *ACM SIGOPS Operating Systems Review*, pp. 15-30, 2001.

[17] M. D. Schroeder, A. D. Birrell, R. M. Needham, "Experience with Grapevine: the growth of a distributed system", *ACM Transactions on Computer Systems*, 2(1):3-23, Feb 1984.

[18] S. H. Strogatz, "Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering", Perseus Book Group, First Edition, 2001.

[19] S.-H. Wu, S. A. Smolka, E. W. Stark, "Composition and behaviors of probabilistic I/O automata", *Theoretical Computer Science*, 176:1-2, pp/ 1-38, Apr 1997.

[20] H. Yu, A. Vahdat, "The costs and limits of availability for replicated services", *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP 01)*, pp. 29-42, Banff, Canada, Oct. 2001.